2207/9862
PATENT

UNITED STATES PATENT APPLICATION
FOR

METHOD AND APPARATUS FOR PREVENTING STARVATION IN A MULTI-NODE ARCHITECTURE

INVENTOR:

Manoj Khare
Akhilesh Kumar
Sin Tan

PREPARED BY:

KENYON & KENYON
1500 K ST., N.W.
WASHINGTON, D.C. 20005

(202) 220-4200

METHOD AND APPARATUS FOR PREVENTING STARVATION IN A MULTI-NODE ARCHITECTURE

FIELD OF THE INVENTION

Embodiments of the present invention relate to a computer system having a multi-node computer architecture. In particular, the present invention relates to a method and apparatus for managing the sending of inter-node messages in a multi-node architecture.

BACKGROUND

Computer systems may contain multiple processors that may work together to perform a task. For example, a computer system may contain four processors that may share system resources (e.g., input devices or memory devices) and may perform parallel processing. The processors may send messages to each other, may send messages to system resources, and may send and receive messages from the system resources (e.g., a memory or output device). For example, such messages may include requests for information that is stored at a memory location in a memory device.

In some systems, the messages are sent from a processor to another component of the system over an interconnect. For various reasons, a message that is sent by a processor may not be received by the destination component. For example, the destination component may not have the capacity to accept the message at that time. Where the sending of a message failed, the processor that sent the message may receive a failure message or may simply not receive an acknowledgment message within a defined waiting period. The message may be in a starvation situation if the message sending fails indefinitely due to the conditions of the system (e.g., the amount of other traffic being directed at the receiving component). In some systems, the processor that is sending the message may alleviate such a starvation condition by taking control of the interconnect until the message is successfully sent.

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a partial block diagram of a system having a node that sends messages according to an embodiment of the present invention.

FIG. 2 is a partial block diagram showing more details of a node controller and receiving agent according to an embodiment of the present invention.

FIG. 3 is a flow diagram of a method of sending messages according to an embodiment of the present invention.

DETAILED DESCRIPTION

Embodiments of the present invention relate to methods and apparatus of sending messages from a node in a multi-node system to a receiving agent. FIG. 1 is a partial block diagram of a system having a node that sends messages according to an embodiment of the present invention. FIG. 1 shows a system 100 which is a computer system that includes processors, memory devices, and input/output devices. Components in system 100 are arranged into architectural units that are referred to herein as nodes. Each node may contain one or more processors, memories, or input/output devices. In addition, the components within a node may be connected to other components in that node though one or more busses or lines. Each node in system 100 has a node connection that may be used by the components within that node to communicate with components in other nodes. In one embodiment, the node connection for a particular node is used for any communication from a component within that node to another node. In system 100, the node connection for each node is connected to a receiving agent 140. A system that has multiple nodes is referred to as a multi-node system. A multi-node system for which each node communicates to other nodes though a dedicated connection may be said to have a point-to-point architecture.

The nodes in system 100 may send messages (e.g., a request to read from a memory) that are directed to a processor or resource in another node. For various reasons, the intended recipient of a message may not be able to process the message or may not receive the message. For example, the recipient of a message may not have room to store a message that it receives. If the sending of the message is not successful, the sending node may retry sending the message. In an embodiment of the present invention, after a sending node determines that a message has been unsuccessfully sent to a receiving agent more than a threshold number of times, the sending node ensures that the message is successfully sent before accepting any new messages to be sent. Use of this embodiment may reduce the length of time which a message waits to be sent and received successfully.

As shown in FIG. 1, system 100 includes a first node 110, a second node 120, a third node 130, and an input/output node 150. Each of these nodes is coupled to receiving agent 140. The term "coupled" encompasses a direct connection, an indirect connection, an indirect communication, etc. First node 110 is coupled to receiving agent 140 through first node interconnect 142. First node interconnect 142 may be one or more lines capable of communicating information to and from the first node.

First node 110 includes processor 111, processor 112, and node controller 115, which are coupled to each other by bus 113. Processor 111 and processor 112 may be any micro-processors that are capable of processing instructions, such as for example a processor in the INTEL PENTIUM family of processors. Bus 113 may be a shared bus. First node 110 also contains a memory 119 which is coupled to node controller 115. Memory 119 may be a Random Access Memory (RAM). Node controller 115 contains an outbound buffer 117 and an outbound buffer manager 118. Outbound buffer 117 may be a memory or part of a memory which stores outbound messages that are generated by a processor, such as processor 111 or processor 112, and are to be sent out from node 110 to another node in system 100. Outbound buffer manager 118 may contain logic that manages the sending of outbound messages that are stored in outbound buffer 117. For example, outbound buffer manager 118 may be an application specific integrated circuit. As used in this application, "logic" may include hardware logic, such as circuits that are wired to perform operations, or program logic, such as firmware that performs operations.

Similarly, second node 120 contains a processor 121, processor 122, processor 123, and node controller 125, each of which is coupled to each other. Second node 120 also contains a memory 129 that is coupled to node controller 125. Third node 130 contains a processor 131, processor 132, and node controller 135 that are coupled to each other. Third node 130 also contains a memory 139 that is coupled to node controller 135. Processors 121, 122, 123, 131, and 132 may be similar to processors 111 and 112. In an embodiment, two or more of processors 111, 112, 121, 122, 123, 131, and 132 are capable of processing a program in parallel. Node controllers 125 and 135 may be similar to node controller 115, and memory 129 and 139 may be similar to memory 119. Processor 131 may contain a cache 133, and processor 132 may contain a cache 134. Cache 133 and cache 134 may be Level 2 (L2) cache memories that are comprised of Static Random Access Memory (SRAM).

In one embodiment, receiving agent 140 may be a routing switch for routing messages within system 100. Receiving agent 140 contains an inbound buffer 147, an outbound buffer 148, and a routing manager 145. Inbound buffer 147 may be a memory, or part of a memory, which stores outbound messages that are received by receiving agent 140. Inbound buffer manager 148 may contain logic that manages the receiving of outbound messages that are stored in inbound buffer 117. Routing agent 145 may contain logic that routes messages received by receiving agent 140 to the appropriate destination node or nodes. Inbound buffer manager 148 and routing manager 145 may be, for example, parts of the same application specific integrated circuit.

Input/output node 150 contains an input/output hub 151 that is coupled to one or more input/output devices 152. Input/output devices 152 may be, for example, any combination of one or more of a printer, keyboard, mouse, or any other input/output device. Input/output hub 151 may by an integrated circuit that contains bus interface logic for interfacing with a bus that complies to the Peripheral Component Interconnect standard. Input/output hub 150 may be similar to, for example, the INTEL 82801AA I/O Controller Hub.

In an embodiment, node controller 115, receiving agent 140, and input/output hub 151 may be a chipset the provides the core functionality of a motherboard, such as a modified version of a chipset in the INTEL 815 family of chipsets

In an embodiment of the present invention, the processors in nodes 110, 120 and 130 may be shared memory multi-processors, and each of the memories 119, 129, and 139 may be part of the same shared physical address space. In a further embodiment, the processors in nodes 110, 120, and 130 communicate with each other through shared memory reads and writes (i.e., by writing to and reading from memory 119, 129 and 139). In a further embodiment, the processors in nodes 110, 120 and 130 each have one or more caches (e.g., Level 1 and Level 2 caches), and these caches are kept coherent using the receiving agent 140. For example, when processor 111 accesses a location in memory 119, it may send a snoop request for that memory location to receiving agent 140, which may determine if any of the processors in second node 120 and third node 130 have cached that memory location. A snoop request may be generated when a processor needs other processors in the system to look in their own caches to see if a particular line is present in their cache.

In a further embodiment, inter-node communication in system 100 is asynchronous (i.e., there is no fixed timing between events). In a still further embodiment, inter-node communication is sent in the form of packets which may contain a header or a header and data sections.

An example of a message size may be 144 bits. In an embodiment, the messages sent may include requests and responses. In a further embodiment, the types of requests that the nodes may send and receive may include a memory read request, memory write request, cache snoop request, cache flush request, memory update request, cache line replacement request, input/output port read request, and input/output port write request. Requests may contain fields such as a packet type, destination ID, request type, source ID, transaction address, request length, stream ID, and ordering semantics.

FIG. 2 is a partial block diagram showing more details of a node controller and receiving agent according to an embodiment of the present invention. In particular, FIG. 2 shows more details of the node controller 115 and receiving agent 140 of FIG. 1. As shown in FIG. 2, output buffer 117 in node controller 115 contains a first message 211, second message 221, and third message 231. In one embodiment, output buffer 117 has the capacity to contain 32 outbound messages. In an embodiment, each outbound message that is stored in output buffer 117 contains a retry counter. In a further embodiment, the retry counter for each outbound message is four bits long, in which case it is capable of counting from 0 to 15. In addition to output buffer manager 118, node controller 115 also contains an input interface 250 and an output interface 255. Output interface 255 is coupled to outbound buffer 117 and is used to send outbound messages from node controller 115 to receiving agent 140. Input interface 250 is used to receive messages from receiving agent 140.

Receiving agent 140 contains inbound buffer 147, inbound buffer manager 148, and routing manager 145 as discussed above. As shown in FIG. 2, inbound buffer 147 contains entries that may be used to store an outbound message after it is received from node controller 115. In one embodiment, each entry in inbound buffer 147 is large enough to store one outbound message that is received from a node. In a further embodiment, inbound buffer 147 contains eight entries. In another embodiment, inbound buffer 147 has 32 entries.

FIG. 3 is a flow diagram of a method of sending outbound messages according to an embodiment of the present invention. In an embodiment, throughout the operation of the system the node controller in a node receives outbound messages from the processors in that node. For example, node controller 115 may receive outbound messages from processors 111 and 112. The outbound message may be a memory related outbound message, such as a request for data stored in a memory location in a memory in another node, such as memory 129. As another example, the outbound message may be related to snooping a cache memory that is part of a second node. In an embodiment, the node controller stores outbound messages which it receives and are destined for another node in an outbound buffer, such as buffer 117. In the embodiment shown in FIG. 3, the node controller sends the outbound messages one-by-one (301) to a receiving agent over the interconnect that connects the node to the rest of the system. The outbound messages may be selected from the buffer for sending using any selection algorithm, such as a first-in-first-out (FIFO) algorithm. The receiving agent may be the destination node, such as for example in a two-node system, or may be a switch, such as receiving agent 140 in FIG. 1. In one embodiment, the outbound message is sent in a packet format.

In an embodiment, the sending of outbound messages is asynchronous. In a further embodiment, new outbound messages that need to be sent may be received by the node controller while outbound messages are being sent out by the node controller. That is, the node controller may store new outbound messages in the outbound buffer while it is sending outbound messages from the outbound buffer to the receiving agent. In an embodiment, because the outbound message may be unsuccessfully sent to the receiving agent, the outbound messages in the outbound buffer of the sending node remain in the buffer after they have been sent to the receiving agent. According to this embodiment, an outbound message may be removed from the outbound buffer after it has been determined that the message has been successfully sent to the receiving agent. In this embodiment, each outbound message that is stored in the outbound buffer may have a sent flag that is set after the outbound message has been sent, and is unset if it is later determined that the sending was unsuccessful. In a further embodiment, an outbound message may be selected for sending only if its sent flag is set.

According to the embodiment shown in FIG. 3, the node that sent the outbound message receives a retry response for the outbound message from the receiving agent (302). The retry response may be sent, for example, if an inbound buffer in the receiving agent does not have room to store the outbound message that was sent by the sending node. Every entry in the inbound buffer may contain an outbound message that was received but has yet to be processed or otherwise transferred to another node, and thus the inbound buffer may not have any free locations which can be used to store new outbound messages. If the inbound buffer does have room to store the outbound message, the receiving agent may send a confirmation message to the sending node after it receives the outbound message and stores it in the inbound buffer. In an embodiment, an outbound message is considered to be successfully sent if a success confirmation message is received for the outbound message from the receiving agent, and an outbound message is considered to be unsuccessfully sent if a retry response is received for the outbound message from the receiving agent.

In an embodiment, when the sending node receives a retry response for a outbound message which is stored in its outbound buffer, it increments a retry counter that is associated with the outbound message (303). A determination is made whether a threshold number of retry responses have been received for this outbound message (304). In one embodiment, the threshold number of retry responses is sixteen retry responses. Of course, a smaller or larger threshold may be used. If the retry counter has not reached the threshold, the node controller may continue sending outbound messages as discussed above. If the retry counter has reached the threshold, the node controller may determine that all of the outbound messages that are in the buffer have been successfully sent before any other outbound messages are sent to the receiving agent. The node controller may prevent any new entries from being stored in the outbound buffer until all outbound messages currently stored in the outbound buffer have been both sent to a receiving agent and stored in an inbound buffer in the receiving agent.

In the embodiment shown in FIG. 3, when it is determined that the retry counter has reached the threshold, the node controller prevents any new outbound messages from being stored in the outbound buffer (305). The node controller then sends outbound messages from the outbound buffer to the receiving agent (306). The node controller determines whether all of the outbound messages in the outbound buffer have been successfully received by the receiving agent (307), and continues to send the outbound messages stored in the outbound buffer until all the outbound messages have been successfully sent. As discussed above, a outbound message may be considered to be successfully sent when a confirmation message is received from the receiving agent for the outbound message. After all the outbound messages in the outbound buffer have been successfully sent, the node controller may allow new entries to be stored in the outbound buffer (308).

In a further embodiment, after receiving a successfully sent outbound message the receiving agent identifies a second node in the multi-node system that is capable of processing the successfully sent outbound message and sends the successfully sent outbound message to the second node.

The present invention may reduce the length of time which an outbound message waits in the outbound buffer to be sent and received successfully. In certain cases, a starvation situation may arise, and an outbound message waiting in the outbound buffer may continually have to be resent to the receiving agent. For example, when the outbound message is sent, the inbound buffer in the receiving agent may be full, and the receiving agent may send a retry response to the node controller. Before the node controller can resend the message, free entries may become available in the inbound buffer but this room may be filled up with other messages that have been sent by the node controller in the interim. In this case, the resent outbound message will again result in a retry response. This sequence may repeat itself indefinitely. This problem may occur, for example, where the messages are sent asynchronously. When the method and apparatus of the present invention are employed, after a particular outbound message has been retried a threshold number of times, the node controller will limit the set of outbound messages sent to those that are currently in the outbound buffer. The number of outbound messages that can fill the inbound buffer in the receiving agent will be quickly be reduced each time a new outbound message is successfully received. Thus, when the present invention is used, the amount of time is limited that any outbound message will stave due to lack of space of the inbound buffer.

Several embodiments of the present invention are specifically illustrated and/or described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention. For example, while the nodes in FIG. 1 are shown containing two/three processors, a node may contain any number of processors. In one embodiment, a node contains only one processor, and in another embodiment a node contains sixteen processors. As another example, while the nodes in FIG. 1 are connected through receiving agent 140, in another embodiment two nodes may be directly connected to each other. For example, in a system with that has two nodes, the node controller of a first node may be connected to the node controller of a second node. In another embodiment, the node controller for a node may be part of a processor in that node. For example, a first node in a system may only contain one processor, and the node controller for that node may be part of that processor. In addition, a node (e.g., node 110) may contain one or processors and an input/output hub. In still another embodiment, the outbound buffer manager, inbound buffer manager, and/or routing manager are implemented as instructions adapted to be executed by a processor, which may include firmware, microcode, object code, source code, ext.